-- Application developers implement their programs by selecting controls from a menu of control systems and placing each control in the desired location on the form. The properties associated with the control type selected are then defined by the developer in order for the control to behave in the desired fashion. Each control type is implemented by a code module which defines the behavior for that control type. The behavior for the control can be adjusted by setting the values of the properties defined for that control type. For example, the behavior for a text box control type accepts data input into its associated field from the keyboard.--

Replace the paragraph beginning at Page 6, line 12 , with the following rewritten paragraph:

--Moreover, the invention allows a robust integration of data between application programs and the operating system. The user no longer needs to physically perform the data transfer function or to develop custom handler software for the data. Once specified in a form, data is dynamically placed into its destination field on the form. Moreover, the invention performs the data placement without requiring rigid rules and custom controls for each field.--

Replace the paragraph beginning at page 12, line 13, with the following rewritten paragraph:

--The data output objects within object class 104 operate in a similar manner. The core object 108 provides an output 120 in a defined format. Depending upon the output device which the program needs to use, or which is selected by the user, the program directs the output to an appropriate output object, which converts it into a format understood by the corresponding peripheral.--

Replace the paragraph beginning at page 20, line 8, with the following rewritten paragraph:

--Referring now to Figure 7, one implementation of an Input Requestor process 500 is illustrated. First the process 500 constructs the device using InputDevice.InputDevice (step 502). Next, the process 500 constructs a controller using an InputController.InputController function and passing it the device information (step 504). The process 500 then enables the controller using an InputController.Enable function step (506) and obtains a buffer using an InputController.MakeBuffer function (step 508).--

Replace the paragraph beginning at page 20, line 19, with the following rewritten paragraph:

Next, the process 500 determines whether the buffer is ready using an InputBuffer.GetReady function (step 514). From step 514, the process 500 determines whether the buffer is ready in step 516. If not, the process loops back to step 514 until the buffer is ready. From step 516, when a buffer becomes available, the process of Figure 5 proceeds to step 518 where it obtains data from the buffer using an InputBuffer.GetString function. Further, the data is processed in the buffer (step 520). The process of Figure 5 then checks whether additional input is needed (step 522). If so, the process loops back to step 512 to continue requesting input into the buffer. Alternately, the process of Figure 5 exits (step 524). --